ReLU and sigmoidal activation functions

Arnold M. Pretorius[0000−0002−6873−8904], Etienne Barnard[0000−0003−2202−2369], and Marelie H. Davel[0000−0003−3103−5858]

Multilingual Speech Technologies, North-West University, South Africa; and CAIR, South Africa.
{arnold.m.pretorius, etienne.barnard, marelie.davel}@gmail.com

Abstract. The generalization capabilities of deep neural networks are not well understood, and in particular, the influence of activation functions on generalization has received little theoretical attention. Phenomena such as vanishing gradients, node saturation and network sparsity have been identified as possible factors when comparing different activation functions [1]. We investigate these factors using fully connected feedforward networks on two standard benchmark problems, and find that the most salient differences between networks with sigmoidal and ReLU activations relate to the way that class-distinctive information is propagated through a network.

Keywords: Non-linear activation function · Generalization · Activation distribution · Sparsity

1 Introduction

The simplest class of neural networks are the multilayer perceptrons (MLPs) [3], which consist of an input layer, one or more hidden layers and an output layer. Each node in a previous layer is connected to all the nodes in the following layer by a vector of weight values. These weight values are adjusted in the learning process so that the network output approximates the labeled target value, minimizing some loss function. To create a non-linear representation and allow the network to learn complex non-linear problems, each node is followed by an activation function that effectively "squishes" or rectifies the output of the node. Two historically popular activation functions for deep neural networks are the established sigmoidal function and the widely used rectified linear unit (ReLU) [1, 13]. Various other activation functions have been proposed, but none are clearly superior to these functions; we therefore investigate only these two functions.

Although several researchers have compared the performance of non-linear activation functions in deep models [1, 2, 12] and the respective difficulties of training DNNs with these activation functions have been established, a more concrete understanding of their effect on the training and generalization process is lacking. It is now widely understood that ReLU networks [1, 6, 13] are easy to optimize because of their similarity to linear units, apart from ReLU units outputting zero across half of their domain. This fact allows the gradient of a rectified linear unit to remain not only large, but also constant whenever the unit is active. A drawback of using ReLUs, however, is that they cannot learn via gradient-based methods when a node output is 0 or less, since there is no gradient [3].

Prior to the introduction of ReLUs, most DNNs used activation functions called logistic sigmoid activations or hyperbolic tangent activations. Sigmoidal units saturate across most of their domain: they saturate to a value of 1 when the input is large and positive, and saturate to a value of 0 when the input is large and negative [2]. The fact that a sigmoidal unit saturates over most of its domain can make gradient-based learning difficult [3]. The gradient of an unscaled sigmoidal function is always less than 1 and tempers off to 0 when saturating. This causes a "vanishing gradients" problem when training deep networks that use these activation functions, because fractions get multiplied over several layers and gradients end up nearing zero.

Thus, each of the popular activation functions faces certain difficulties during training, and remedies have been developed to cope with these challenges. The general consensus is that ReLU activations are empirically preferable to sigmoidal units [1, 6, 12, 13], but the evidence in this regard is not overwhelming and theoretical motivations for their superiority are weak. These theoretical motivations focus heavily on the beneficial effects of sparse representations in hidden layers [1], without much evidence of its effect on network training and generalization performance or the behavior of hidden nodes. Another strong theme in previous comparisons of activation functions is that of "vanishing gradients", and although this phenomenon is visible and pertinent in many cases, any discrepancies in generalization performance cannot fully be attributed to vanishing gradients when sigmoidal networks are able to train to 100% classification accuracy. Overall, we suspect that there are more factors that need to be considered when comparing activation functions, specifically regarding the behavior of hidden nodes. In this study we therefore revisit the training and generalization performance of DNNs trained with ReLU and sigmoid activation functions. We then investigate the effect of the activation function on the behavior of nodes in hidden layers and how, for each of these functions, class information is separated and propagated. We conclude by discussing the broader implications of these findings.

2 Comparing performance

We compare the performance of DNNs of varying width and varying depth on several tasks. The two main activation functions of interest are the rectified linear unit (ReLU) and the sigmoidal unit, as motivated in Section 1. We investigate fully connected feed-forward neural networks on the MNIST [9] and CIFAR10 [8] datasets – convolutional networks are more commonly used on these tasks and achieve higher accuracies, but we limit our attention to fully connected networks in order to investigate the essential components of generalization in DNNs. MNIST is a relatively simple dataset that contains 60 000 images of handwritten digits ranging from 0 to 9, while CIFAR10 is a more complex dataset that contains 60 000 images of 10 different objects, including: planes, ships, cats and dogs. For each dataset we split the data into a training, validation and test set. We trained several networks, as summarized below:

– Network depths of: 2, 4, 6 and 8 layers.
– Networks widths of: 200 and 800 nodes.
– With batch-normalization layers and without batch-normalization layers.

We optimize the hyper-parameters that most strongly affect the model convergence and generalization error, namely the learning rate and training seed. Values for hyper-parameters such as the scale of initial random weights and layer biases are selected after performing several initial experiments to determine the combinations of hyper-parameters that give good convergence with high validation accuracy. The optimizer used to train the neural networks is Adam [7], due to its adaptive estimates of lower-order moments compared to normal stochastic gradient descent. In all experiments, we optimize over three different random seeds when searching for hyper-parameter values to increase our degree of certainty that these values are acceptable. We use an iterative grid search to determine appropriate learning rate values, and let the learning rate decay with a factor of 0.99 after every epoch.

We let all models train for 300 epochs. To regularize the network training, early stopping is used. No other explicit regularization is added to the networks, such as L1 and L2 norm penalties. We do however acknowledge the regularizing effect of batch normalization [5]. Cross-entropy is used as loss function with a softmax layer at the end of the network. We use Xavier [2] initialization for sigmoid networks and He [4] initialization for ReLU networks. We ensure that networks are sufficiently trained by tracking the training loss over epochs as well as the training and validation accuracy. We also compare the performance of our models to benchmark results for MNIST [10, 14] and CIFAR10 [11] to ensure that our results are comparable. On the MNIST dataset, similar architectures reach a test accuracy of 98.4%, while much larger networks than those shown in this paper reach 56.84% on the CIFAR10 dataset.

Figure 1 shows an example of the average learning and validation curves over three seeds with standard error; results shown here are for networks trained on the MNIST dataset. We observe that for different architecture configurations, the training curve of the ReLU and sigmoid networks are similar while the validation curves of the ReLU networks show better performance overall.

The learning curves from Figure 1 are obtained on the training and validation sets. The models performing best on the validation set are evaluated on the test set. Evaluation accuracy on the test set is reported over different experiments in Figure 2. From this figure we see that the ReLU networks generally perform better than the sigmoid networks on the evaluation set. However, when the network is wide enough, the performances of sigmoid and ReLU networks become more similar when not using batch normalization. An interesting observation is that when the sigmoid network has a constant width of 200 nodes, the application of batch normalization results in an increase in performance with an increase in

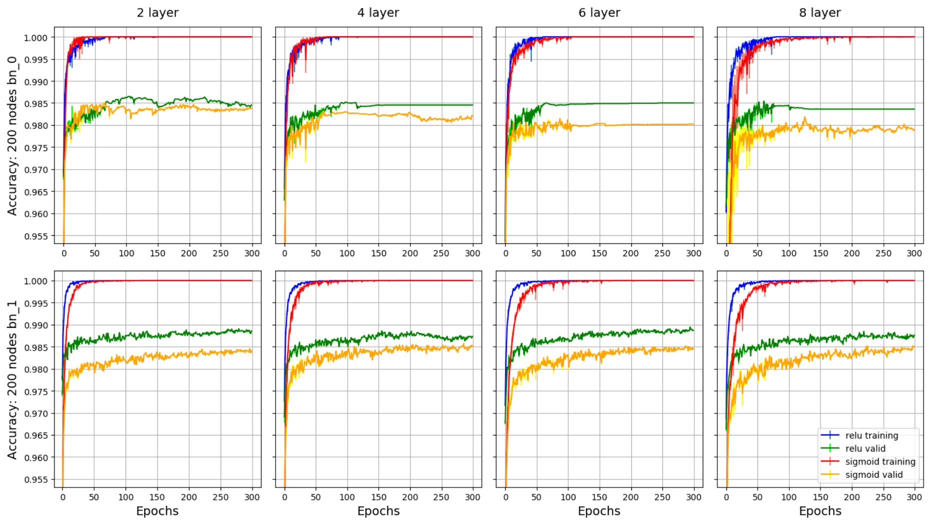

Fig. 1. Comparison of training and validation curves of ReLU and sigmoid networks with increase in depth (left to right) and a fixed width of 200 nodes. The top row of networks are trained without batch normalization while the bottom row is trained with batch normalization. (MNIST)

depth. When not using batch normalization, the generalization performance of the sigmoid networks decreases with an increase in depth.

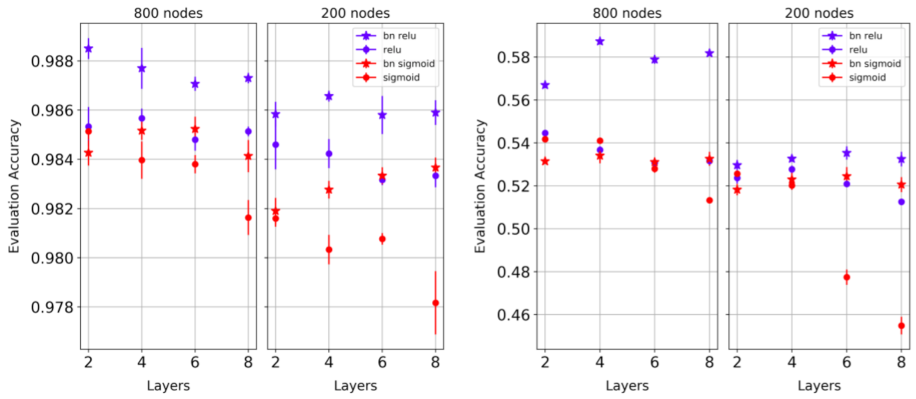

Fig. 2. Evaluation accuracy averaged over seeds for networks with 200 and 800 width trained on (a) MNIST and (b) CIFAR10. Starred points represent networks that are trained using batch normalization.

In the case of CIFAR10 from Figure 2(b), the ReLU networks only reliably generalize better than the sigmoid networks when trained with batch normalization. When not trained with batch normalization, the ReLU and sigmoid networks generalize less well with an increase in depth. The sigmoid networks perform similarly to ReLU networks in all configurations where batch normalization is not used, except for the 6- and 8-layer networks that are 200 nodes wide. The sigmoid networks generalize relatively poorly with these two configurations compared to the other configurations. This poor generalization could be attributed to the vanishing gradient problem since these two network architectures struggle to fit the training set. In contrast, we see a general increase in evaluation accuracy when increasing the number of hidden layers while training with batch normalization.

In summary, then, we observe that optimal generalization is obtained when batch normalization is used to train wide ReLU networks; for CIFAR10, network depth provides a small additional benefit. In contrast to [2] we cannot ascribe these benefits to the vanishing gradients problem, since all our networks, apart from the 6- and 8-layer sigmoid networks from Figure 2(b), can train to virtually perfect classification of the training set. An alternative explanation is therefore required, and the next section investigates a number of clues related to such an explanation.

3 Node distributions

To better understand the effect of the activation functions on training and generalization performance, we investigate the behavior of nodes after the activation function is applied. We specifically look at the activation values for each class at each node to investigate how class information is separated and propagated throughout the network. The term "activation distribution" is used to refer to the distribution of activation values for the samples of a class at a hidden node. We choose to show results only for the 4-layer networks as they have moderate depth and similar trends are observed in deeper and wider networks.

At each hidden node $h_{i,j}$ we calculate the class activation distributions $(a_{c,i,j})_{x_{c,1}}^{x_{c,M}}$ for each class c after applying the activation function T so that for a single sample-node pair we get:

$$a_{c_m,i,j} = T(\sum_{k=0}^{s_{(i-1)}} w_{i,j,k} h_{i-1,k}) \qquad 1 \leq i \leq N \tag{1}$$

with sample $x_{c,m}$ as input, and the distribution as:

$$(a_{c,i,j})_{x_{c,1}}^{x_{c,M}} = \{a_{c_1,i,j}, a_{c_2,i,j}, a_{c_m,i,j}, ..., a_{c_M,i,j}\} \quad 1 \leq i \leq N \quad 1 \leq c \leq C \tag{2}$$

where c is the class index, i the layer index, j the node index and s the number of nodes in a layer. $x_{c,1}$ is the first sample in a class c and $x_{c,M}$ the last sample with C the number of classes. For each class activation distribution $(a_{c,i,j})_{x_{c,1}}^{x_{c,M}}$ at each node we calculate the median and standard deviation of the distribution.

When training a DNN with non-linear activation functions, there exists points where the learning process is saturated and gradients reach zero or near-zero values, depending on the activation function. For ReLU this saturation point is at 0.0, where values equal to or below this point have zero gradient and learning subsides for that specific sample at that node. The sigmoid function has two of these saturation points, at 0.0 and 1.0 respectively. For the sigmoid function, gradients never reach zero when nearing these saturation points, but they become very small and any weight update with this gradient has almost no effect. The position of the median with regard to the saturation points, and with regard to medians from activation distributions of other classes at a node provides an indication of how the class information is separated among different classes. The standard deviation gives an indication of the variability of the activation values and and the extent to which the majority of activation values approach the saturation points.

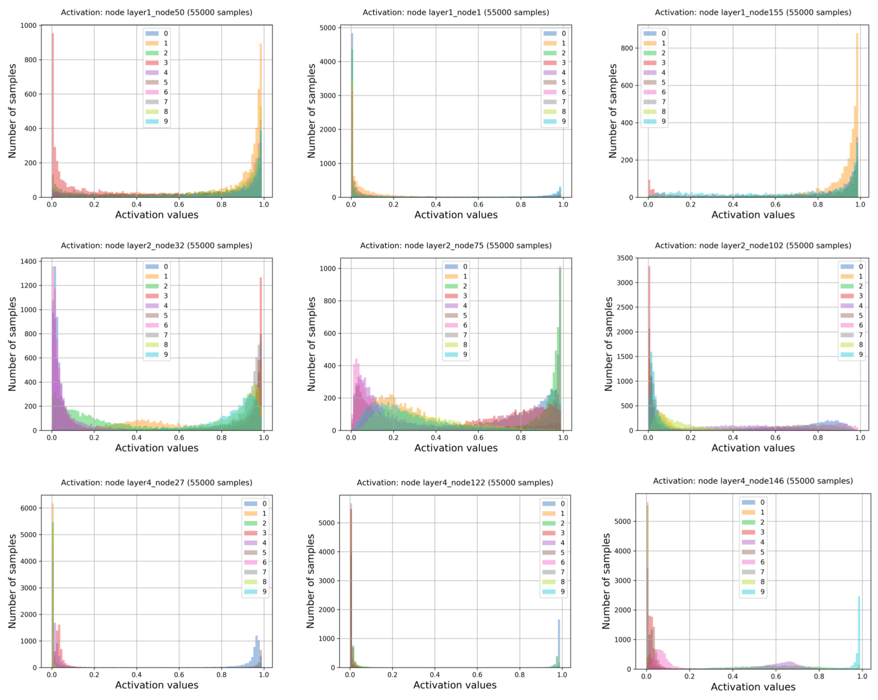

Fig. 3. Activation distributions of generic nodes in shallow (top row), intermediate (middle row) and deeper (bottom row) layers after sigmoid activation function is applied. (MNIST)

Figure 3 shows the typical output of specific hidden nodes in the trained network after the sigmoid activation function has been applied to the nodes. We can see that the activation distributions are highly non-uniform and are skewed towards the 0.0 and 1.0 saturation points. Interestingly the outputs of the second hidden layer seems to be less saturated with higher variance. (We observe this phenomenon at the second layer for all deeper networks as well.) The bottom row shows typical activation distributions at nodes in the last hidden layer. These distributions are heavily skewed towards the saturation points, with some distributions almost being point distributions with very low variance. We observe that when class information is separated, activation distributions still overlap each other near the saturation points.

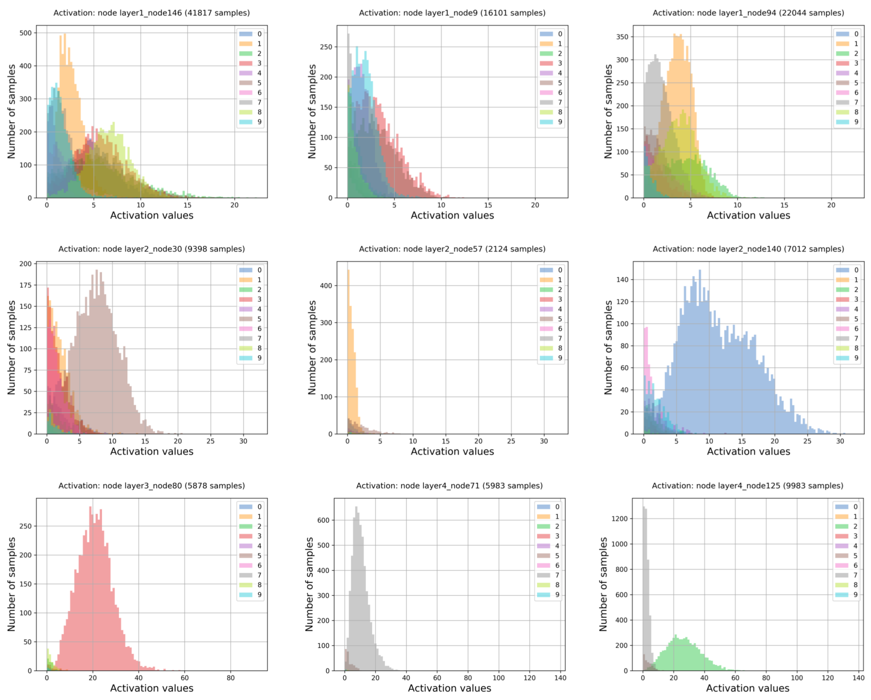

Fig. 4. Activation distributions of generic nodes in shallow (top row), intermediate (middle row) and deeper (bottom row) layers after ReLU activation function is applied. (MNIST)

Figure 4 shows the typical output of specific hidden nodes in the trained network after the ReLU activation function has been applied at the node. It is clear that the activation distributions are cut off at 0.0 since activation values below 0.0 are rectified while the positive values retain their magnitude. The distributions in the positive domain retain some of the same uniform shape, depending on how many samples have negative activation values. In the positive domain, the activation distributions are not saturated and do not overlap as heavily for different classes such as the output of nodes in Figure 3. Not only do the nodes in deeper layers become more specialized towards one class, they activate for fewer samples than the nodes in earlier layers. This confirms the ability of ReLU trained networks to be sparse.

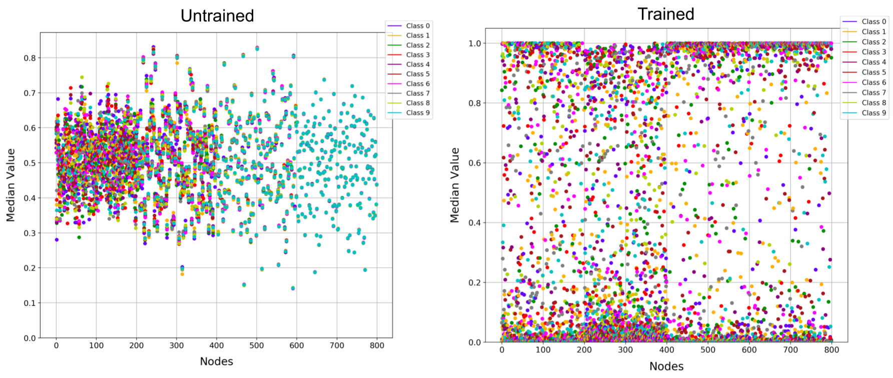

Fig. 5. Sigmoid: Median values of activation distributions for each class at every node in a hidden layer for the untrained (left) and trained (right) model. (MNIST)

To understand how class information is separated and propagated, we plot the median of the activation distributions of each class at each node. Figures 5 through 7 show the median values for the distributions after the activation function has been applied. The x-axis indexes the nodes in the networks. For a 4x200 network, this means that nodes 1-200 make up layer one, nodes 201-400 layer two, etc. Since the activation distributions of the sigmoid network are highly non-uniform, we track the position of activation distributions by plotting the median of the distributions. By using the median values, we regard the distribution as a whole and not only the location of the distribution mass.

For the untrained model, Figure 5 shows that the medians are mostly clustered around 0.5 for the first hidden layer. The medians in the second hidden layer (nodes 200-400) are slightly more saturated towards 1.0 and 0.0 and the median values are starting to overlap more for classes at the same node. From nodes 400 to 600 it is observed that the medians of the distributions overlap significantly, meaning that with the untrained weights the nodes are not able to distinguish activation values of one class from the others. In the last hidden layer (nodes 600-800) it is observed that the activation distributions lay almost directly on top of each other and class information cannot be separated. From Figure 5 we see that when the model is sufficiently trained, class information is separated by saturating medians towards 0.0 and 1.0. This effect is more clear at deeper hidden layers (nodes 400-599 and 600-800), while more nodes in earlier layers have activation distributions with median values slightly closer to 0.5.

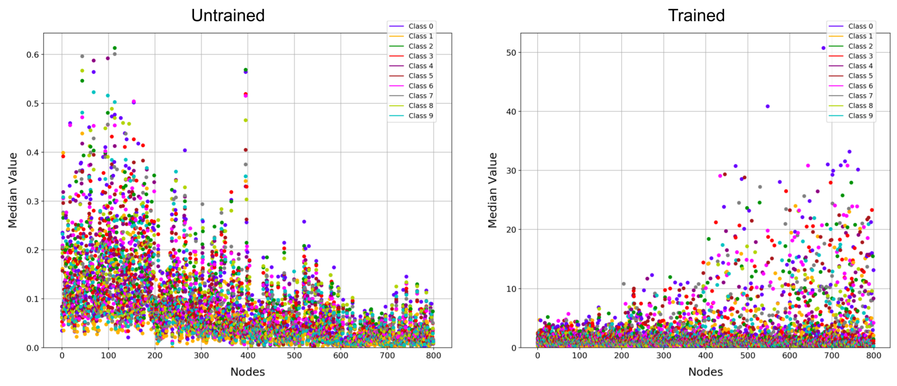

Fig. 6. ReLU: Median values of activation distributions for each class at every node in a hidden layer for the untrained (left) and trained (right) model. (MNIST)

For ReLU networks the mean and the median are almost identical due to the uniform shape of their activation distributions. Figure 6 shows the untrained model for the network trained with ReLU activation functions. The median values of node distributions are less saturated in earlier layers and saturate more towards 0.0 for deeper layers. It is important to note that the activation distributions in deeper layers do not overlap as strongly compared to the sigmoid network in Figure 5. The ability of the ReLU network to completely suppress class information and not activate for specific samples allows the network to still separate distributions of different classes in deeper layers. Even when distributions are not separated in a meaningful/learned way, the inherent "sparse" structure that the ReLU activations introduce suggests better separation of class information.

From Figure 6 it is observed that when a ReLU network is sufficiently well trained, the activation distributions in earlier layers have lower activation values and class information is suppressed with overlap of activation distributions of classes. This observation with the behavior seen in Figure 4 indicates that nodes in earlier layers remain relatively agnostic towards classes. The nodes in deeper layers have less overlap and nodes become more specialized towards specific classes.

Figure 7 shows the median values of activation distributions of networks trained on the CIFAR10 dataset. If the network cannot effectively fit the more complex data, class information cannot be effectively separated and the median values look more similar to the untrained models in Figure 5 and Figure 6.

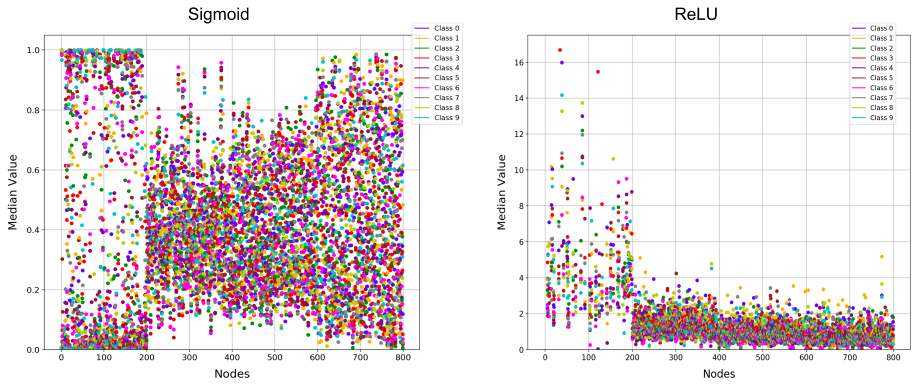

Fig. 7. CIFAR10: Median values of activation distributions for each class at every node in a hidden layer for networks trained with sigmoid (left) and ReLU (right) activations.

Table 1. Layer sparsity for networks trained with ReLU activations without and with batch normalization. Sparsity here refers to the average percentage of samples per node that are inactive for a hidden layer.

	ReLU		**ReLU BN**	
Layer	**Untrained**	**Trained**	**Untrained**	**Trained**
MNIST (%)				
1	49.22	80.40	49.08	65.76
2	48.85	79.33	49.70	56.67
3	48.45	81.93	49.81	68.77
4	48.74	81.90	49.15	69.12
CIFAR10 (%)				
1	51.07	77.56	51.06	58.53
2	50.96	76.55	50.75	59.41
3	50.73	77.54	50.95	57.43
4	50.85	77.14	50.81	57.37

Since the beneficial effects of sparsity in deep rectifier models have been proposed as a mechanism for their performance [1], we calculate the sparsity of one of our models trained with and without batch normalization. In Table 1 we compare the sparsity of each layer for these two networks. The metric is determined by counting the number of samples that are inactive for each node and then averaging over nodes to get the sparsity metric for the layer. Networks that are trained with batch normalization have fewer sparse interactions per layer, meaning that nodes are (on average) active for more samples of a class. We know however from Figure 2(a) and 2(b) that the use of batch normalization causes the generalization performance of the ReLU networks to increase. This leads us to believe that sparsity, although useful as proven by Glorot et al. [1], does not entirely describe the impressive generalization abilities of ReLU networks.

4 Conclusion

In this study we compared the performance of different neural network architectures trained with ReLU and sigmoid activation functions. We observed that when networks are wide enough, sigmoid networks have comparable generalization performance to ReLU networks. The ReLU networks benefit more from the use of batch normalization than networks trained with sigmoid activations.

We investigated and compared the behavior of nodes trained with these two activation functions by looking at how class information is separated. It was observed that while ReLU and sigmoid networks both are able to effectively separate class information, they do this in fundamentally different ways. The sigmoid networks saturate class information towards 0.0 and 1.0 where there is plenty of overlap between activation distributions. The ReLU networks saturate class information towards 0.0 for earlier layers with moderate overlap of class distributions, making earlier layers more conservative towards class discrimination while nodes in later layers become more specialized towards single classes. We also show that when training a ReLU network with batch normalization, the hidden layers have lower average sparsity but superior generalization performance compared to ReLU networks trained without batch normalization.

Overall we find that sparsity and saturation seem less pertinent than the way in which class-distinctive information is propagated through the networks, and how node behavior differs. We plan to relate the node behavior of DNNs trained with different activation functions to their generalization performance in future studies.

Acknowledgments

This work was partially supported by the National Research Foundation (NRF, Grant Number 109243). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and the NRF does not accept any liability in this regard.

References

1. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 315–323. PMLR, Fort Lauderdale, FL, USA (11–13 Apr 2011), http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf
2. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (13–15 May 2010)

3. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
4. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. CoRR **abs/1502.01852** (2015), http://arxiv.org/abs/1502.01852
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR **abs/1502.03167** (2015), http://arxiv.org/abs/1502.03167
6. Jarrett, K., Kavukcuoglu, K., LeCun, Y., Ranzato, M.: What is the best multi-stage architecture for object recognition? ICCV'09 pp. 16,24,27,173,192,226,363,364,525 (2009). https://doi.org/10.1109/ICCV.2009.5459469, http://yann.lecun.com/exdb/publis/pdf/jarrett-iccv-09.pdf
7. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (12 2014)
8. Krizhevsky, A., Nair, V., Hinton, G.: CIFAR10 dataset, https://www.cs.toronto.edu/ kriz/cifar.html
9. LeCun, Y., Cortes, C., Burges, C.: MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/
10. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
11. Lin, Z., Memisevic, R., Konda, K.R.: How far can we go without convolution: Improving fully-connected networks. CoRR **abs/1511.02580** (2015), http://arxiv.org/abs/1511.02580
12. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models (2013)
13. Nair, V., Hinton, G.: Rectified linear units improve Restricted Boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. pp. 807–814. ICML'10 (2010), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.6419
14. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: International Conference on Document Analysis and Recognition (ICDAR). vol. 02, p. 958 (08 2003)